Summer 2012

MIXED REALITY ROBOTICS

Pleo Tool Kit – Programming in C++

University of Southern California

Professor Shiela Tejada

Elisabeth Brooks

Yekaterina Glazko

Eric Kapitanski

Irina Tyshkevich

MIXED REALITY ROBOTICS

Table of Contents

The Pleo

About the Pleo

Pleo is an animatronic robot designed to model the hypothetical behaviors of a baby camarasaurus. Some features of the Pleo include the ability to react to new situations and different types of stimuli, the ability to progressively develop new traits as the Pleo engages in more and more interaction, and the life-like ability to express different emotions and behaviors. The Pleo was designed by Caleb Chung, the co-creator of Furby, Chung's company Ugobe sold Pleo and was manufactured by Jetta until 2009, when Ugobe filed for bankruptcy and Jetta re-launched the Pleo line under their own name.

The Pleo robot has camera-based vision for light detection and navigation, two microphones that can detect sound directionally under quiet conditions, twelve touch sensors that make Pleo responsive to contact, four foot sensors for surface detection, fourteen force-feedback sensors (one on each joint), an orientation tilt sensor so that the Pleo can recognize when it is not upright, an infrared sensor for object detection in the mouth, the ability to communicate with other Pleos through infrared two-way, and infrared detection for external objects. These hardware components provide a realistic and highly interactive experience for the user with Pleo.

Created initially as a children's toy, Pleo exhibits behaviors that are often pleasing to children: Pleo often initiates play and cuddling sessions, and isn't difficult to bring back into a positive state if made upset. The range of behaviors for Pleo, however, is only as large as Pleo's states of varying emotion, and it includes playfulness, contentedness, sadness, fear, and frustration.

By using artificial intelligence and sensor input to interpret emotion, Pleo will respond accurately to his situation. For example, if Pleo is left alone for a prolonged period of time, he may walk forward and make attention-seeking noises. If Pleo is tired, he may put himself into a sleep position and snooze off. One of Pleo's most dramatic emotionally responses comes

from when he feels that he might be in danger: or namely, hanging upside-down by the tail. We challenge the reader to find out for himself how Pleo will respond.

Pleo's producing company, Ugobe, understood the potential benefits of being able to use Pleo for development in robotics. Due to Pleo's large amount of sensors and detecting mechanisms, the ability to create new behaviors for the Pleo isn't an obscure challenge- and Ugobe is supportive of developers thanks to the publication of a very comprehensive programming guide and a developer resource center on their website. To write new behaviors for Pleo, one needs to familiarize themselves with Pawn and the individual sensors so that they can put together new movements and reactions for Pleo. Pleo comes with an SD Card slot on the underside of his stomach, and new behaviors and actions can be written on the SD Card in Pawn, and inserted for Pleo to execute these behaviors.

The Pleo is a very useful tool for learning robotics or expanding one's programming horizons. Just to be able to learn and analyze all of Pleo's current behaviors and how they are executed via artificial intelligence and movement is a feat of its own, the ability to write one's own is a step towards understanding how to create an accurate and entertaining interaction between the user and the robot. Pleo is a robot that requires time and one's participation to fully get to know, but the realistic, and even fun, nature of the interaction will make that time and participation invested an appealing investment.

Past Projects

Schuyler Middle School: A comprehensive lesson plan for middle-school students and developing for Pleo.
http://www.digitalwish.com/dw/digitalwish/view_lesson_plans?id=6361
http://teacherweb.com/NE/SchuylerGradeWW/DPrescott/apt22.aspx

Georgia Tech: A curriculum developed for the Pleo and understanding robotics and perception. They have their own wiki relating to using XBee and MiGio.
http://ipr10.wikidot.com/pleo

ICE Distance Education: A collection of projects that students can do with Pleo in the Pawn language, ranging from beginner to advanced levels of difficulty.
http://ice-web.cc.gatech.edu/dl/?q=node/573

Hampton University: A research paper describing how Pleo was used to assist children with cerebral palsy learn how to dance.
Wii Nunchuk Controlled Dance Pleo! Dance! to Assist Children with ...

Pleo Development Kit: The ideal starting place for anyone hoping to begin developing for the Pleo robot.
http://www.pleoworld.com/eng/pdk.php

Resources

Bob the Pleo: A discussion forum and website of compiled resources in regards to modifying and "hacking" the Pleo.
http://bobthepleo.com/forums/index.php?topic=732.0

Lirec: Hack your Pleo and connect it to a game that is played on your Android phone using a bluetooth module.
http://www.lirec.eu/pleohack

Pleo Dinosaur Life: A good start for overall information regarding the Pleo robot, its hardware capabilities, and general behaviors.
http://pleodinosaurlife.blogspot.com

PleoWorld: Articles and blogs pertaining to new developments with Pleo.
http://www.pleoworld.com/pleo_rb/eng/news.php

Pleo Wiki: A comprehensive compilation of papers regarding Pleo's use in research. A good place to start for research and project ideas.
http://pleo.wikia.com/wiki/Pleo_in_Research

Understanding Pleo's Hardware

Connecting to the Pleo via Terminal

It is possible to connect to the Pleo via USB and issue him instructions in real time via Terminal. With this method, it is also possible to monitor and activate the various sensors. These interactions occur while Pleo is operating in normal mode, and seem to be more like "suggestions" than commands. For example, if you ask Pleo to play a sound while he is asleep, he will refuse.

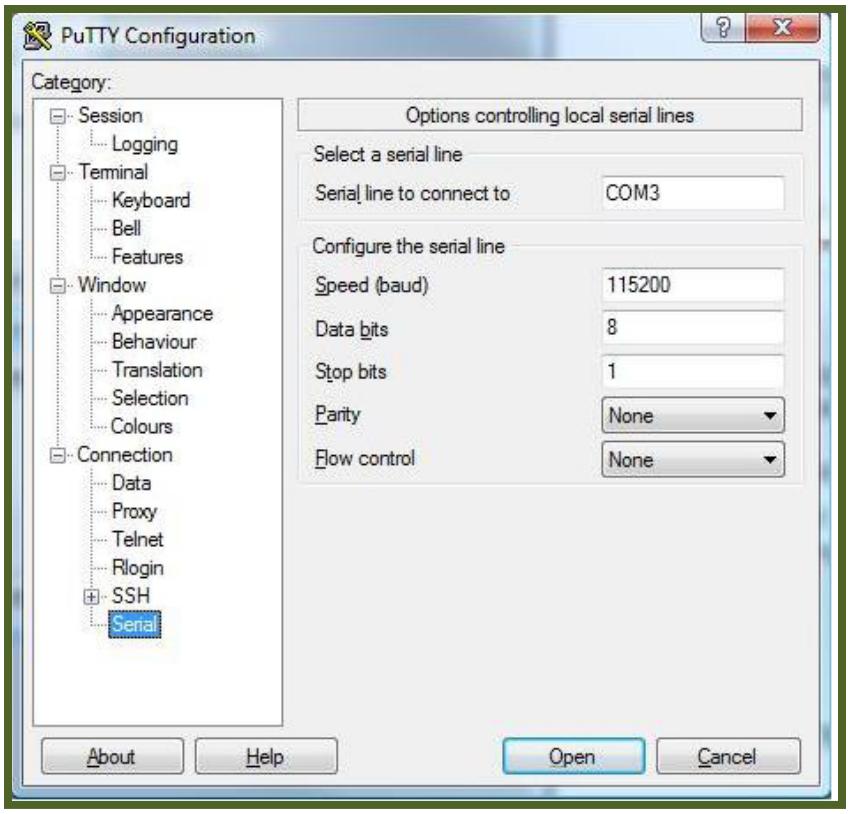
There is a key distinction between this real time commanding and using a routine on the SD card – like, for example using a compiled routine from MySkit (described in the Motors and Joins section) – because in that mode you have Pleo's "full attention" from the start and he is not running his pre-programmed behaviors. Contrarily, when communicating with Pleo in real time, the commands entered in via Terminal will be competing with his normal programming.

There exist some accounts of success combing the two methods (such as making a "blank" routine on the SD card so Pleo is not running LifeOS, and then issue him commands in real time while he is not doing anything else). It should also be noted that the USB connection is rather unreliable and has the tendency to randomly terminate.

Windows 7 (32 bit)

On the Windows Operating System, the easiest way to connect to the Pleo is via PuTTY and USB. In order to do this, it is necessary to download the Pleo Development Kit from PleoWorld.com or here **http://ipr10.wikidot.com/pleo** - this Kit has the necessary driver which allows the Pleo to connect to the computer.

While the Pleo is turned off, connect the USB port to the computer. Turn the Pleo on and wait for the computer to recognize that the driver for the connection is not yet installed. Manually direct the Pleo to install the correct driver which can be found in the PleoDevelopmentKIt/tools/drivers. Once the driver is installed, check which COM port the Pleo is connected to in "Devices".



In the Serial Category, the settings should match that of the above diagram. In Session, choose the "Serial" option and then open the Terminal.

Mac OS

To connect to the Pleo via Terminal on Mac, connect the USB to the computer. Turn on the Pleo and then open the Terminal. Type in " **ls/dev/tty.***" This will bring up a list of devices connected to the computer. Look for the one that contains *.usbmodem* followed by a *fa141* or *fd131*. Then type in " **screen/dev/tty.usbmodemfd131 115200**" (or *fa141* if that was on the previous screen). This is what the screen should look like:

```
guest-wireless-207-151-073-165:~ lizzbrooks$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.LGGR500-SerialPort1
/dev/tty.Bluetooth-PDA-Sync     /dev/tty.usbmodemfa141
guest-wireless-207-151-073-165:~ lizzbrooks$ screen /dev/tty.usbmodemfa141 115200
```

Commands and Parsing Feedback

Once in the Terminal, you have access to most of Pleo's sensors. The command "***sensor show***" will open up a list of all the sensors and preset values. Further exploration of the "***help***" command will explain who to alter and change these values.

This is what the sensor **show screen** will look like:

```
> sensor show
                Aux   Debounce
        Trig   Trig     Time                  Trig
 Sensor Level  Level   in ms.  Enabled Trig  Time   TmSince   Count       Value  RawValue  Name
      2    25      0        0        1     0   50491   44891     3 (  2%)     75        75 SENSOR_BATTERY
      7     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_IR_ACTIVITY
      8     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_HEAD
      9     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_CHIN
     10     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_BACK
     11     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_LEFT_LEG
     12     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_RIGHT_LEG
     13     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_LEFT_ARM
     14     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_RIGHT_ARM
     15     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_ARSE
     16     0      0        0        1     0   21867    5074     4 (  3%)      0         0 SENSOR_FRONT_LEFT
     17     0      0        0        1     0   16949    9293     8 (  7%)      0         0 SENSOR_FRONT_RIGHT
     18     0      0        0        1     0   21929   13885     5 (  4%)      1         1 SENSOR_BACK_LEFT
     19     0      0        0        1     0    6468    6468     1 (  0%)      1         1 SENSOR_BACK_RIGHT
     20     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_CARD_DETECT
     21     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_WRITE_PROTECT
     22    40     30        0        1     0    2265    2265     1 (  0%)     56        56 SENSOR_LEFT_LOUD
     23   150     30        0        1     0    9530    1210     4 (  3%)     10        10 SENSOR_LIGHT
     24    40     30        0        1     0    2265    2265     1 (  0%)     52        52 SENSOR_RIGHT_LOUD
     25    40     10        0        1     0    9065    2263     2 (  1%)      0         0 SENSOR_OBJECT
     26     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_MOUTH
     28     1      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_SOUND_DIR
     29    30    -30        0        1     0       0       0     0 (  0%)     -1        -1 SENSOR_LIGHT_CHANGE
     30    50     40        0        1     0    2265    2265     1 (  0%)     52        52 SENSOR_SOUND_LOUD
     31     0      0        0        1     0    8843    6437     3 (  2%)      1         1 SENSOR_TILT
     32    -1      0        0        1     0   62441      31    66 ( 65%)      0         0 SENSOR_TERMINAL
     34     0      0        0        1     0    1913    1913     1 (  0%)      1         1 SENSOR_USB_DETECT
     35     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_WAKEUP
     36    55     51        0        1     0       0       0     0 (  0%)     25        25 SENSOR_BATTERY_TEMP
     38   150     75        0        1     0    5800    5800     1 (  0%)      0         0 SENSOR_SHAKE
     39    35    -35        0        1     0       0       0     0 (  0%)     16        16 SENSOR_LOUD_CHANGE
     40     0      0        0        1     0       0       0     0 (  0%)      0         0 SENSOR_BEACON
     41     0      0        0        1     0       0       0     0 (  0%)    314       314 SENSOR_BATTERY_CURRENT
>
```

For example, this row here shows the Battery sensor information. The Trigger level is 25, because the robot will shut down if the battery value is below 25%. This battery, however, is currently at a value of 75%.

The most important columns in this chart are:
“Sensor” – which states the sensor number
“Enabled” – which states whether the sensor is on “1” or off “0”
“Value” – which states the current set value of the sensor, especially important for battery
“Name” – which states the name of the sensor

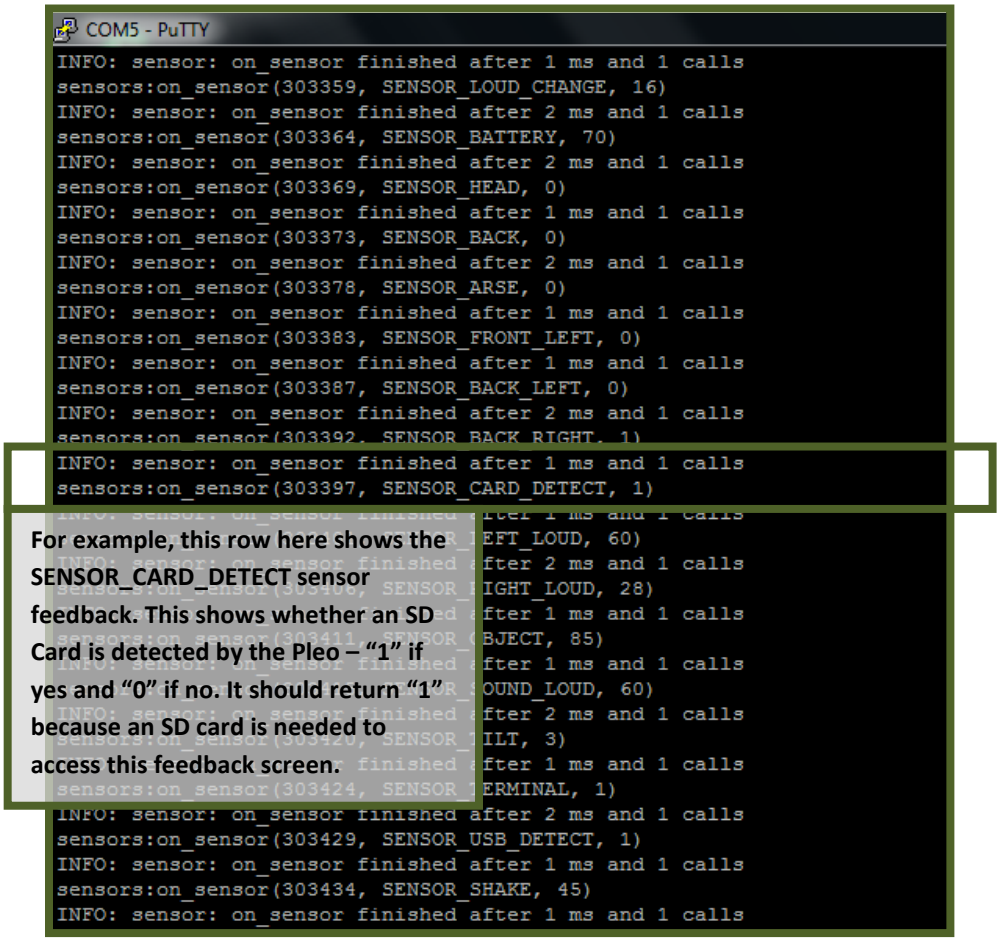
The rest of the information, such as “Trig Level” – which is the set value at which the sensor is triggered – can be taken into account, but aren’t necessary important to programming the Pleo.

However, just gaining access to the sensors does not provide real time feedback as to what sensors the Pleo is currently feeling. In order to do this, a PAWN program must be written, compiled onto an SD card and then inserted into the Pleo. This PAWN program commands the Pleo to return sensor feedback to the Terminal screen.

The Pleo Development Tool Kit that is needed to install the appropriate driver for Pleo to connect to a computer also comes with many “.p” PAWN programs that can be used to control the Pleo and access information. These files are a key part to getting sensor data from the Pleo. In order to print sensor data onto the screen, the only thing that must be altered is “sensors.p” file which can be found in the Tool Kit in the folder called bin. Replace that program code with this code and then compile it onto an SD card:

```
// Very simple sensors.p example. Add code to on_sensor for those
// sensors you would like to respond to.

// save space by packing all strings
#pragma pack 1

#include "Log.inc"
#include "Script.inc"
            #include "Sensor.inc"
#include "Sound.inc"

#include "joints.inc"
#include "Joint.inc"
#include "Motion.inc"
#include "motions.inc"
#include "sounds.inc"


public init()
{
print("sensors:init() enter\n");
print("sensors:init() exit\n");
}

public on_sensor(time, sensor_name: sensor, value)
{
new name[32];
sensor_get_name(sensor, name);
printf("sensors:on_sensor(%d, %s, %d)\n", time, name, value);
}

public close()
{
print("sensors:close() enter\n");
print("sensors:close() exit\n");
}
```

The sensor feedback looks like a constant stream of sensor information. In order to pause the stream press “control” + C:



Sensor Evaluation

Most of Pleo's sensors fall into one of five categories: touch, motor, sound, camera and infrared. The most reliable and useful of these are the touch and motor sensors.

Infrared Sensors play a large role in leaf and other Pleo detection, but they are very difficult to access and understand. Moreover, the infrared readings are rarely reliable.

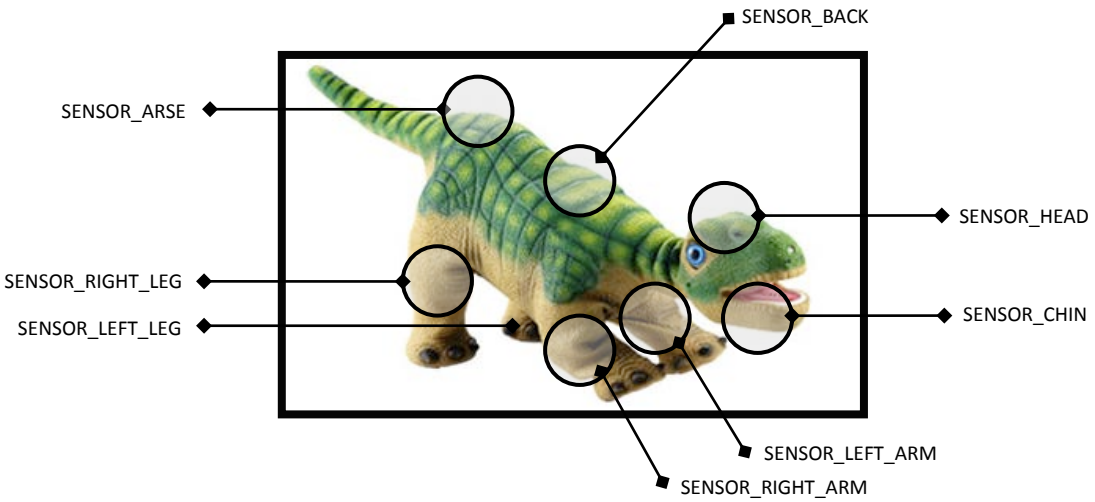
Pleo's camera can take photos and has some blob detection. However, this camera is very limited because it cannot receive real time images (all images are saved to the SD card) and the blob detection is incredibly unreliable.

The robot's speakers allow the behaviors to be complemented by emotion evoking sounds, which can be very useful. But, as far as sound detection goes, the Pleo does not recognize sounds very well because of high sound wave interference from the environment.

Motors and Touch sensors, however, work well together in creating emotion evoking actions and behaviors that can be implemented into robot education of elementary school children.

A sensor that will not aid much in programming, but is important to keep track of is the Battery Sensor complex, which is described in the Miscellaneous Sensor category.

Touch Sensors



The Pleo comes equipped with eight capacitive touch sensors that seamlessly work together with the skin so it doesn't feel like you're pressing a button, but rather just interacting with the robot. They look like thin, metal strips and are located on the robots back, legs, shoulder, head, and chin.

These sensors detect proximity of objects (not touch or pressure), and give the robot a sense of their outside environment, in what way they are being handled and how they should send signals to the motors to move in reaction to the detected input. Ground touch sensors on the bottom of the feet let the Pleo know if it has been picked up, or if it is standing on a solid surface.

A little more technically speaking, the Pleo's touch sensors are made of foil patches that measure the charge accumulation upon detecting proximity. The transfer of charge between the circuit and the ground plane can then trigger a signal to the Pleo's microcontroller.

Terminal Command: to gain access to and change touch sensor values…

sensor help

Relevant Senor: SENSOR_HEAD
This sensor reflects the state of touch the touch sensor under the Pleo's head. Sensor feedback states "1" if there is currently something within the proximity of the head (the head is being touched) and "0" if there is nothing near it. In the example below, SENSOR_HEAD shows a reading of "0", so the Pleo's head is not currently being touched.

```
INFO: sensor: on_sensor finished after 2 ms and 1 calls
sensors:on_sensor(303448, SENSOR_HEAD, 0)
```

Relevant Senor: SENSOR_CHIN
This sensor reflects the state of the touch sensor under the skin on the Pleo's chin. Sensor feedback states "1" if there is currently something within the proximity of the chin (the chin is being touched) and "0" if there is nothing near it.

Relevant Senor: SENSOR_BACK
This sensor reflects the state of the touch sensor under the skin on the Pleo's upper back, near the neck. Sensor feedback states "1" if there is currently something within the proximity of the back (the back is being touched) and "0" if there is nothing near it. Generally, the back touch sensors will be activated without physical contact by the mere movement of something near the sensor.

Relevant Senor: SENSOR_ARSE
This sensor reflects the state of the touch sensor under the skin on the Pleo's arse. Sensor feedback states "1" if there is currently something within the proximity of the arse (the arse is being touched) and "0" if there is nothing near it. The arse sensor does not necessarily require a heavy touch to be activated, but mere proximity is usually not enough.

Relevant Senor: SENSOR_LEFT_LEG
This sensor reflects the state of the touch sensor under the skin on the Pleo's upper left leg. Sensor feedback states "1" if there is currently something within the proximity of the left leg (the leg is being touched) and "0" if there is nothing near it.

Relevant Senor: SENSOR_RIGHT_LEG
This sensor reflects the state of the touch sensor under the skin on the Pleo's upper right leg. Sensor feedback states "1" if there is currently something within the proximity of the right leg (the leg is being touched) and "0" if there is nothing near it.

Relevant Senor: SENSOR_LEFT_ARM
This sensor reflects the state of the touch sensor under the skin on the Pleo's upper left arm.

Sensor feedback states “1” if there is currently something within the proximity of the left arm (the arm is being touched) and “0” if there is nothing near it.

Relevant Senor: SENSOR_RIGHT_ARM
This sensor reflects the state of the touch sensor under the skin on the Pleo’s upper right arm. Sensor feedback states “1” if there is currently something within the proximity of the right arm (the arm is being touched) and “0” if there is nothing near it.

Relevant (Ground) Sensors: SENSOR_FRONT_LEFT, SENSOR_FRONT_RIGHT, SENSOR_BACK_LEFT, SENSOR_BACK_RIGHT
These ground sensors are located on the bottom of Pleo’s feet. They are in an activated state of value “1”, when the foot is standing on a surface. The value of the sensor will be “0” if Pleo’s foot is in the air. For example, the sensor reading below shows that the value of SENSOR_BACK_RIGHT is “1”, meaning that the back right foot is touching the ground.

```
INFO: sensor: on_sensor finished after 2 ms and 1 calls
sensors:on_sensor(303392, SENSOR_BACK_RIGHT, 1)
```

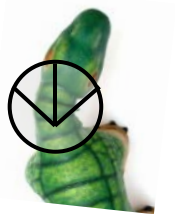
Motors and Joints

There are 14 motors located all over the Pleo. These motors give the robot the ability to walk, wag its tail and crane its neck. All of these motors have force feedback sensors, so they are able to detect the environment surrounding the Pleo.

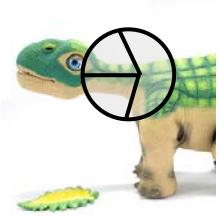
All of the motors already exist in the Pleo and the robot comes pre-programmed with some natural movements that the Pleo does on its own in a natural environment. However, because the motors move wires in the robot in response to instructions from processors, it is also possible to program more complex and unique movements that utilize these motors.

Pleo's motors allow the robot to generate purposeful actions which often relay emotion. These movements can be in response to touch or a stimulus. Alternatively, Pleos can be programmed to do movements without any trigger.

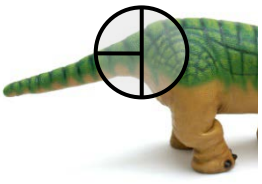
Head
This motor is at neutral position, when the Pleo is looking straight ahead. Otherwise, the robot can look 90 degrees up and 90 degrees down.



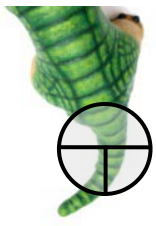
Neck (Horizontal)
The neutral position is straight forward. The neck can, also, move 65 degrees left and right.



Neck (Vertical)
This motor is at neutral position, when the Pleo is looking straight ahead. The neck can, also, move 75 degrees up or down.



Tail (Vertical)
Neutral position is straight back. Otherwise, the tail can move 90 degrees up and 90 degrees down.



Tail (Horizontal)
Neutral position is straight back. Otherwise the tail can move 90 left and 90 degrees right.

Shoulders (2)
Straight is the neutral position for this motor. It can also move 55 degrees forward from the neutral and state and 20 degrees backwards.

Elbows (2)
Straight is the neutral position for this motor. The elbow motor can only move 30 degrees forward from the neutral state.

Hips (2)
The neutral state for this motor type is straight down. The motor can, alos, move 45 degrees forwards and backwards.

Torso
The neutral position is straight forward. The motor can move 35 degrees left and right.

Knee (2)
The neutral state for this motor is straight down. The motor can also rotate 50 degrees backwards from the neutral state.

Terminal Command: in order to control the movements of Pleo enter:

help joint

This command will give access to more ways of commanding the Pleo. For example, if **joint neutral** is entered into the terminal, the Pleo will move all of its joints to neutral position. The terminal will return the progress of each joint during the process of returning to neutral position.

```
> joint neutral
j0 a23 17928 ms J_ST_RUNNING (1)
j0 a23 17960 ms J_ST_SLOW (5)
j0 a2 18600 ms J_ST_SLOW (5)
Joint moved to 2 in 672 ms
j1 a58 18600 ms J_ST_RUNNING (1)
j1 a34 19145 ms J_ST_SLOW (5)
j1 a2 20777 ms J_ST_SLOW (5)
Joint moved to 2 in 2177 ms
j2 a-30 20777 ms J_ST_RUNNING (1)
j2 a-30 20810 ms J_ST_SLOW (5)
j2 a-27 20970 ms J_ST_RUNNING (1)
j2 a-24 21034 ms J_ST_SLOW (5)
j2 a-2 21674 ms J_ST_STOPPED (2)
```

Relevant Sensor: SENSOR_TILT

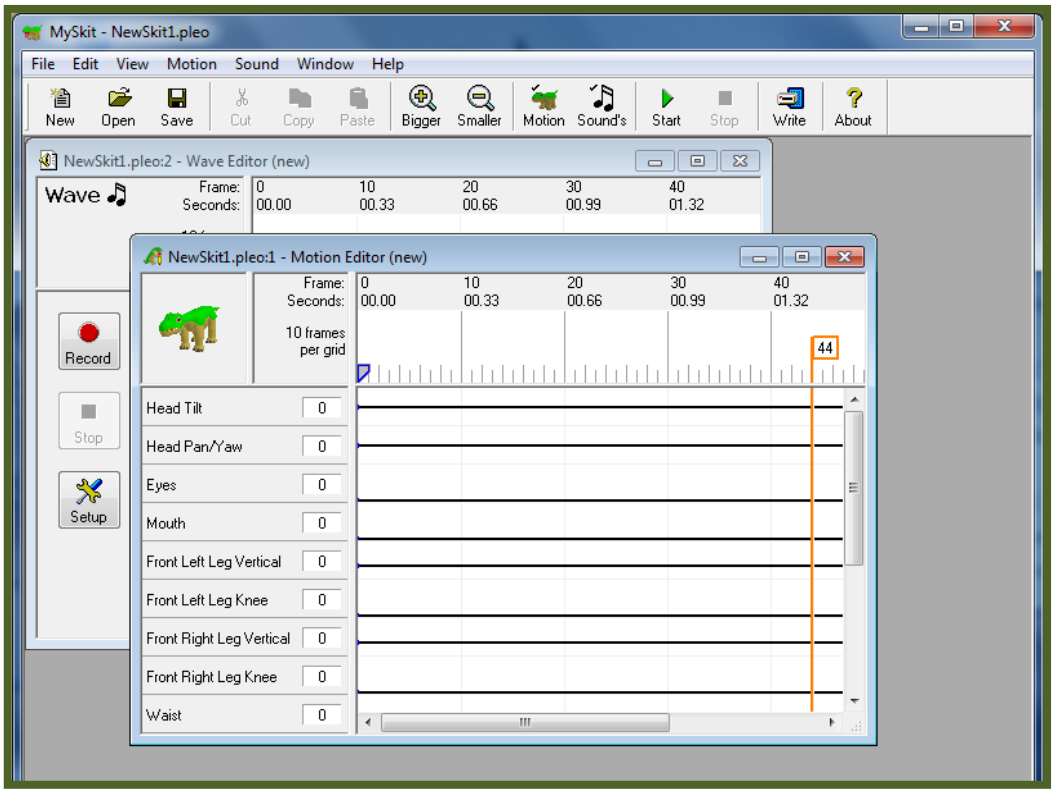
This sensor detects the orientation of Pleo's torso in three spaces and is triggered when the title sensor moves into a new position. The sensor can return these values…

TILT_NONE = 0 (no orientation known)
TILT_ON_FEET = 1 (feet are oriented downwards with respect to torso)
TILT_LEFT_SIDE = 2 (on left side)
TILT_RIGHT_SIDE = 3 (on right side)
TILT_ON_NOSE = 4 (front of torso is pointed downwards)
TILT_ON_TAIL = 5 (aft-end of torso is pointed upwards)
TILT_ON_BACK = 6 (feet are pointed upwards with respect to torso)

Relevant Sensor: SENSOR_SHAKE

This sensor is used to detect if the Pleo is being shaken, like, for example, when it is being woken up. The value of this sensor can be between 0 and 255. The sensor triggers when the shake frequency goes from below 75 to above 150.

Another easy way to control and create new movements for the Pleo is through a program called *MySkit*. This is very easy to use, and is basically a GUI that lets you set a routine of sound and movement for the Pleo. You then "compile" your skit and upload it to a SD card which you plug into the Pleo. When starting up, the Pleo first checks the SD card for any programs before it boots into normal routines.



Motions can be activated by connecting to Pleo via a terminal and typing the following command:

motion play (Motion number)

So, for example, the code for walk forward is 8330. To make Pleo walk forward, enter "**motion play 8330**" into the terminal and the robot will walk forward. For a list of all available motions, type the command "**motion show**".

Emotions that Pleo Motions evoke:

Happy

motion ID 8320 = com_tail_wag
motion ID 8321 = com_tail_wag_v2
motion ID 8322 = com_tilt_back_tickle_la
motion ID 8323 = com_tilt_back_tickle_ll
motion ID 8324 = com_tilt_back_tickle_ra
motion ID 8325 = com_tilt_back_tickle_rl
motion ID 8343 = emo_act_happy_v1
motion ID 8344 = emo_act_happy_v2
motion ID 8407 = happy_honk
motion ID 8354 = emo_fidget_happy_v1
motion ID 8355 = emo_fidget_happy_v2

Sad

motion ID 8467 = soc_abuse_whimper
motion ID 8345 = emo_act_sad_v1
motion ID 8346 = emo_act_sad_v2
motion ID 8393 = fat_nap_fidget_pain
motion ID 8394 = fat_nap_fidget_sad
motion ID 8395 = fat_nap_fidget_scared
motion ID 8264 = com_disappointed_v1
motion ID 8265 = com_disappointed_v2
motion ID 8266 = com_fallen_l_breath_v1
motion ID 8267 = com_fallen_l_breath_v2
motion ID 8268 = com_fallen_tilt_l_in
motion ID 8269 = com_fallen_tilt_l_loop_v1
motion ID 8270 = com_fallen_tilt_l_loop_v2
motion ID 8271 = com_fallen_tilt_l_sleep
motion ID 8272 = com_fallen_tilt_r_in
motion ID 8273 = com_fallen_tilt_r_loop_v1
motion ID 8274 = com_fallen_tilt_r_loop_v2
motion ID 8275 = com_fallen_tilt_r_sleep
motion ID 8356 = emo_fidget_pain_v1
motion ID 8357 = emo_fidget_sad_v1
motion ID 8358 = emo_fidget_scared_v1
motion ID 8347 = emo_act_scared_v1
motion ID 8348 = emo_act_scared_v2

Tired

motion ID 8482 = soc_rest_fidget_v1
motion ID 8483 = soc_rest_fidget_v2
motion ID 8484 = soc_rest_fidget_v3
motion ID 8485 = soc_rest_fidget_v4
motion ID 8486 = soc_rest_laydown
motion ID 8487 = soc_rest_listen
motion ID 8488 = soc_rest_pant
motion ID 8489 = soc_rest_pant_v1
motion ID 8490 = soc_rest_pose_v1
motion ID 8491 = soc_rest_rxn_head
motion ID 8492 = soc_rest_rxn_tail
motion ID 8493 = soc_rest_sniff
motion ID 8566 = yawn_3
motion ID 8396 = fat_nap_fidget_sniff
motion ID 8397 = fat_nap_fidget_sound
motion ID 8398 = fat_nap_fidget_v1
motion ID 8399 = fat_nap_fidget_v2
motion ID 8400 = fat_nap_getup
motion ID 8401 = fat_nap_getup_v2
motion ID 8402 = fat_nap_getup_v3
motion ID 8403 = fat_nap_lay_down
motion ID 8404 = fat_sleep_lay
motion ID 8405 = fat_tired
motion ID 8406 = fat_yawn

Angry

motion ID 8337 = emo_act_angry_v1
motion ID 8338 = emo_act_angry_v2
motion ID 8350 = emo_fidget_angry_v1
motion ID 8351 = emo_fidget_angry_v2
motion ID 8462 = picked_up_shake
motion ID 8463 = picked_up_squirming
motion ID 8464 = singing_howling
motion ID 8465 = soc_abuse_head_chin
motion ID 8466 = soc_abuse_rxn
motion ID 8565 = upside_down
motion ID 8388 = fat_nap_fidget_angry
motion ID 8364 = exp_object_react_growl

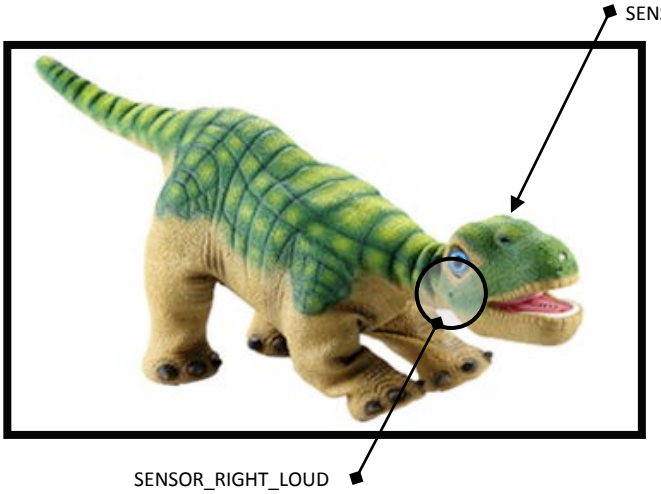
Hungry

motion ID 8412 = hun_baby_bird_feeding
motion ID 8413 = hun_beg
motion ID 8414 = hun_bite_s
motion ID 8415 = hun_chew_drop
motion ID 8416 = hun_chew_drop_stuck
motion ID 8417 = hun_chew_fast
motion ID 8418 = hun_chew_slow
motion ID 8419 = hun_chew_v1
motion ID 8420 = hun_dropped_sniff
motion ID 8421 = hun_exit
motion ID 8422 = hun_expecting
motion ID 8423 = hun_fidget_scratch_left
motion ID 8424 = hun_graze_big_bites
motion ID 8425 = hun_graze_drink
motion ID 8426 = hun_graze_ripping
motion ID 8427 = hun_graze_stand
motion ID 8428 = hun_graze_v1
motion ID 8429 = hun_graze_v2
motion ID 8430 = hun_graze_v3
motion ID 8431 = hun_happy_graze
motion ID 8432 = hun_happy_honk
motion ID 8433 = hun_hatch_bird
motion ID 8434 = hun_hatch_bird_whimper
motion ID 8435 = hun_hatch_cry_down
motion ID 8436 = hun_hatch_cry_inward
motion ID 8437 = hun_hatch_cry_up
motion ID 8438 = hun_hatch_tantrum_A
motion ID 8439 = hun_hatch_tantrum_B
motion ID 8440 = hun_lip_smack
motion ID 8441 = hun_moo
motion ID 8442 = hun_paw_ground_l
motion ID 8443 = hun_paw_ground_r
motion ID 8444 = hun_rxn_back
motion ID 8445 = hun_rxn_chin
motion ID 8446 = hun_rxn_head
motion ID 8447 = hun_rxn_la
motion ID 8448 = hun_rxn_ra
motion ID 8449 = hun_search
motion ID 8450 = hun_sniff_l
motion ID 8451 = hun_sniff_r
motion ID 8452 = hun_sniff_s
motion ID 8453 = hun_sniff_stand_ground
motion ID 8454 = hun_sniff_walk_air
motion ID 8455 = hun_stand
motion ID 8456 = hun_stand_chewing_cud
motion ID 8457 = hun_stand_looking
motion ID 8458 = hun_stand_sniff_ground
motion ID 8459 = hun_tantrum_for_food
motion ID 8460 = hun_tummy_rumble
motion ID 8461 = hungry_cry

Miscellaneous

motion ID 8349 = emo_fidget_affect_v1
motion ID 8352 = emo_fidget_bored_v1
motion ID 8353 = emo_fidget_curious_v1
motion ID 8359 = exp_in
motion ID 8360 = exp_object_gone
motion ID 8361 = exp_object_react_bark
motion ID 8362 = exp_object_react_bite
motion ID 8363 = exp_object_react_curious
motion ID 8339 = emo_act_bored_v1
motion ID 8340 = emo_act_bored_v2
motion ID 8341 = emo_act_curious_v1
motion ID 8342 = emo_act_curious_v2
motion ID 8408 = hatch_coax_walk1
motion ID 8409 = hatch_coax_walk2
motion ID 8410 = hatch_reward
motion ID 8411 = hiccup
motion ID 8276 = com_fidget_cough
motion ID 8277 = com_fidget_sneeze
motion ID 8278 = com_hatch_twitchy_stand
motion ID 8279 = com_head_held_v1
motion ID 8280 = com_hello_bark_lg
motion ID 8281 = com_hello_bark_sm
motion ID 8282 = com_hello_howl
motion ID 8283 = com_hello_playfight
motion ID 8284 = com_holding_fidget_l_v1
motion ID 8285 = com_holding_fidget_l_v2
motion ID 8290 = com_holding_l
motion ID 8291 = com_holding_pre_l
motion ID 8292 = com_holding_r
motion ID 8293 = com_light_off_v1
motion ID 8294 = com_light_on_v1
motion ID 8295 = com_listen_s
motion ID 8296 = com_obj_detect_l
motion ID 8297 = com_obj_detect_r
motion ID 8298 = com_obj_detect_s
motion ID 8299 = com_puke
motion ID 8300 = com_rxn_buck
motion ID 8301 = com_rxn_buttup
motion ID 8302 = com_rxn_dizzy
motion ID 8303 = com_rxn_howl
motion ID 8304 = com_rxn_lite_off
motion ID 8305 = com_rxn_lite_on
motion ID 8306 = com_rxn_tickle
motion ID 8307 = com_shat
motion ID 8308 = com_sit
motion ID 8309 = com_sneeze_v1
motion ID 8314 = com_sniff_neutral_l_v2
motion ID 8318 = com_tail_held_v1
motion ID 8319 = com_tail_held_v2
(and many more…)

Sound and Speaker Sensors



The Pleo has two microphones on its head – on the right and the left side that appear as small holes - which allow noise to be detected.

Pleo has two speakers – one located in the mouth and on just above the tail. These speakers allow the robot to converse with noise.

Pleo already comes with some pre-programmed sounds. But, you can record your own sounds to play from the Pleo. This can be done through the MySkit Wave sound editor.

Terminal Command: to gain more access to commanding Pleo's sound bank…

help sound

```
> help sound
Syntax: sound <command>
Sound help screen:
  play <num>|<name>    play sound file [sounds/]<num>|<name>.usf|wav.
  show                 show all available sounds
  speed <percent>      Set sound playback speed in percent of normal.
  volume <percent>     Set sound playback volume in percent of normal.
>
```

This command will give access to many different sound commands, sound titles available on the Pleo and ways to adjust the sound aspects.

The sound library has a sound called "moo". For example, if you enter in **play sound moo** into the terminal, the Pleo will make the "moo" sound.

Relevant Sensor: SENSOR_SOUND_DIR
This sensor triggers when a definite sound is detected. It returns the direction in degrees relative to the Pleo where the sound was heard. A value lower than 90 means that the sound was detected to the left of the Pleo and a value greater than 90 means that the sound was detected to the right of the Pleo, while a value of 0 means that the sound was straight ahead. A loud sound with no direction changes the value to -128. Pleo is best at recognizing long sounds, as it gives the sensor more time to find a pattern and decipher the sound.

Relevant Sensor: SENSOR_SOUND_LOUD
This sensor detects the edges of loud sounds that stand out from background sound levels. The sensor triggers when the value of the volume of the sound is greater than “40” or becomes less than “30”. The value range is between 0 and 100.

Relevant Sensor: SENSOR_LOUD_CHANGE
This sensor is used to detect large changes in ambient noises based on the long-term average noise level which is monitored. The sensor value is the difference between the short-term detected noise level and the long term average level. A value between -127 and 127 will be returned by this sensor. In the example below, the Pleo heard a sound that was a value of 16 greater than the average sound level of its current environment.

```
INFO: sensor: on_sensor finished after 1 ms and 1 calls
sensors:on_sensor(303438, SENSOR_LOUD_CHANGE, 16)
```

Relevant Sensor: SENSOR_LEFT_LOUD
This sensor returns the absolute loudness of the left microphone. In the example below, this Pleo is hearing a sound with the volume value of 60 with the left microphone.

```
INFO: sensor: on_sensor finished after 1 ms and 1 calls
sensors:on_sensor(303401, SENSOR_LEFT_LOUD, 60)
```

Relevant Sensor: SENSOR_RIGHT_LOUD
This sensor returns the absolute loudness of the right microphone. In the example, the robot, the same Pleo from the example, above is hearing a sound with the volume of 28 with the right microphone.

```
INFO: sensor: on_sensor finished after 2 ms and 1 calls
sensors:on_sensor(303406, SENSOR_RIGHT_LOUD, 28)
```

Emotions that Pleo Sounds evoke:

Happy	Angry/ Tired	Surprised / Inquisitive	Upset /Pained/ Scared	Action/ Miscellaneous
sq1_purr04	sur_002	sq1_question02	sq1_question01	sq1_sniff01-07
sq1_question03	sur_003	sq1_question04	sq1_sbn01	sq1_snore01-02
sq1_tickle01-11	tired_pant	sq1_surprised01	sq1_scared01	sq1_snort01-05
ta_daa	tired_pant_breathe	sq1_surprised02	sur_004	test_fail
	yawn	sq1_suspicious01	surprised_yipe	test_ok
		sq1_suspiscious02		test_success
		sur_001		

Camera

The Pleo robot comes equipped with a color camera located just above the mouth, on the nose. This allows the Pleo to detect changes in light, different colors and, also, motion.

The on board Pleo camera also takes pictures in QCIF format, which is 176x144 pixels. The image is able to be saved in either .bmp or .raw formats.

Terminal Command: to test the camera and capture an image enter…

camera capture <filename> [bmp|raw] [new|last]

For example, ***camera capture test.bmp*** takes a picture and stores it as "test.bmp". However, the camera is sometimes unreliable and a blank image can occur after issuing the command to take a picture.

Images from the Pleo are saved to the SD card, which means that it is impossible to extract and analyze them in real time. There is also no way to stream live video.

However, it is possible to use PAWN code for blob tracking. In order to do this, the code must first be written in PAWN and then compiled and loaded onto an SD card ahead of time. It would be very challenging to implement any blob tracking or optical character recognition in C++ using the on board camera, as any C++ solution would basically need to "trigger" a PAWN command to track images. It is not possible to pass parameters such as colors or shapes to the PAWN script in real time.

In summary, it is not possible to view real-time camera output from the on board camera in C++. But, the on board camera can definitely be utilized.

Relevant Sensor: SENSOR_LIGHT
This sensor indicates the current absolute ambient light level that the Pleo's camera module detects. The sensor value range is 0-255 and triggers when the value crosses 30 and 150 in either direction - lower values indicate darker areas and higher values indicate light areas.

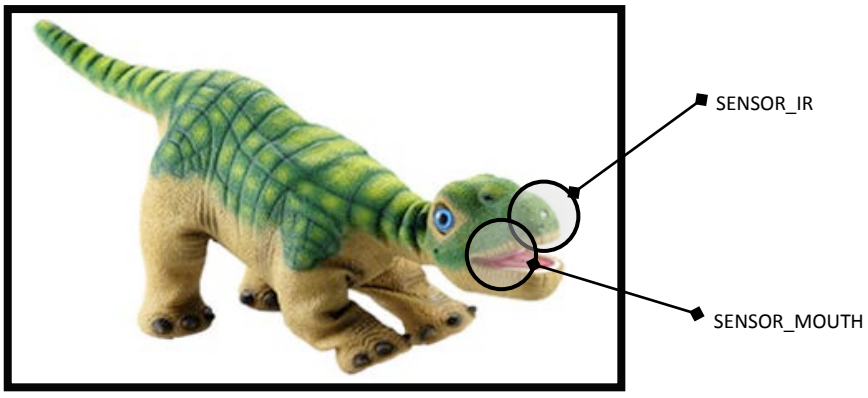
Relevant Sensor: SENSOR_LIGHT_CHANGE
This sensor detects relative changes to the current light level. The value of the sensor can fall between -127 and 127. The sensor is triggered when the value becomes greater than 30 (the light gets brighter) or when the value becomes less than 30 (the light gets darker).

Infrared Sensors

The Pleo has multiple infrared sensors on its body. These sensors allow the robot to detect and interact with other Pleos, leaves and other external objects.

One infrared transmitter and receiver is located on the top of the Pleo's nose. One infrared interrupter is located inside of the Pleos mouth which allows it to detect when an object (for example the leaf) has been placed in the mouth.

These infrared sensors can sense other Pleos as well as objects in the mouth based on whether or not the Pleo has a specialized reaction that it will perform. For example, the infrared sensors in the mouth can sense a leaf and establish a grip. This is how the Pleo can play tug-of-war.



Relevant Sensor: SENSOR_IR
This sensor is used to indicate reception of valued NEC-format IR data that shows something other than another Pleo or a leaf have been detected. The value of this sensor is the number of lines of data in the Infrared receive buffer. The sensor triggers a value of "1" if any data has been received by the infrared sensor and a "0" if nothing has been received.

Relevant Sensor: SENSOR_MOUTH
This sensor indicates the presence or absence of an infrared-opaque object in Pleo's mouth, such as the leaf. This sensor returns a value of "1" when an object is sensed in the mouth and a value of "0" when no objects are sensed in the mouth.

Relevant Sensor: SENSOR_BEACON
This sensor is used to indicate other Pleo's and their interaction and is triggered when a beacon is received from another Pleo. The sensor returns a value of "1" when another Pleo is detected and a value of "0" when another Pleo is not detected.

Relevant Sensor: SENSOR_IR_ACTIVITY
This sensor interprets any code that attempts to program the Pleo's actions and reactions.

Miscellaneous

SENSOR_CARD_DETECT
This sensor indicates the state of the SD Card slot, giving feedback as to whether or not there is an SD Card with a program that will override the Pleo's out-of-the-box code. A value of "1" indicates that an SD Card is inserted, while a value of "0" indicates that there is no SD Card detected.

SENSOR_WHITE_PROTECT
This sensor reflects the state of the write protect switch on the inserted SD Card and is triggered when the card is inserted or removed. If the value is "1" the card is protected; if the value is "0" the card is not protected.

SENSOR_OBJECT
This sensor indicates whether or not there is an object in front of the Pleo. The value of the sensor ranges between 0 and 100 – 100 means there is definitely something and 0 indicates that nothing is there. This sensor does not measure distance, but rather the probability that an object is present, has appeared, or has disappeared.

SENSOR_TERMINAL
This sensor triggers when a line has been entered. The value of the sensor is the number of characters received and the sensor triggers when the user types one or more characters on the monitor interface, including just a return.

SENSOR_USB_DETECT
This sensor reflects the state of the USB port. A value of "1" is returned is a USB is detected and plugged into the computer and a value of "0" is returned if no USC cord is detected.

The following sensors are not very important and are little known about:
SENSOR_RESERVED
SENSOR_CHARGER_STATE

The Battery

SENSOR_BATTERY

This sensor revolves around the state of the battery and returns the current percentage value of the charge level. The battery level is measured on a scale of 0 to 100. However, the robot's performance in relation to the battery level is very unpredictable and unreliable. The battery will rarely charge fully to 100% and often begins to work poorly below 50% - and the amount of time it takes for the battery to die varies greatly. Pleo goes into sleeping position/mode and then shutdown when battery value is below 25%.

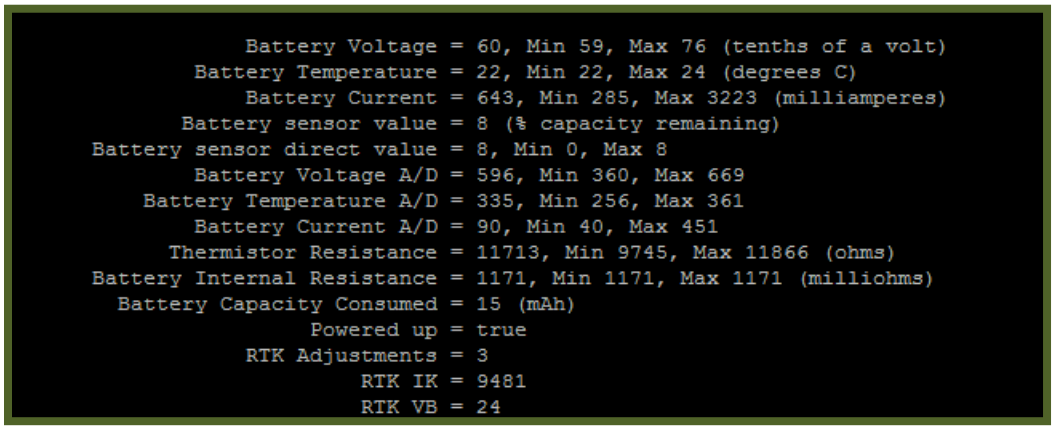
SENSOR_BATTERY_TEMP

This sensor is used to monitor the temperature of the battery in the Pleo robot. Pleo's firmware will automatically shut down if the temperature reaches 58 degrees Celsius.

SENSOR_BATTERY_CURRENT

This sensor returns the number of milliamps being consumed from the battery by all the motors in the Pleo on a scale of 0 to 5000.

Here is an example of sensor feedback received about a Pleo's battery:



Programming with Pleo

C++ Scripting

In order to use C++ with Pleo, you will need to use a Windows operating System.

If you have a 64-bit version of Windows, you will need everything listed below.
If you have a 32-bit version of Windows, you will only need the last three items:

1. VMWare Player
2. Windows 32-bit (in VM Player)
3. Visual Studio 2010
4. Pleo Development Kit
5. SerialGio Project File (Eric has this)

If you are using the 64-bit version of windows, please download VMWare player: https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/4_0
Then you will need a windows 32-bit image. Install VMWare Player and load the 32-bit Windows image onto it. For the rest of this guide you will only be interacting with the Virtual Machine.

Download Visual Studio 2010 and the Pleo Development Kit. Connect Pleo to your computer via USB and turn him on. Windows should detect that there is a new device attached, but it will be unable to find the driver. You will need to open your device manager ("computer" -> "properties" -> "device manager"). Select the item called "Pleo" with the little yellow triangle next to it. Right click on it and go to "properties" -> "driver tab" -> "update driver" -> browse my computer for computer software". Find where you installed the Pleo Development Kit, and go to "tools" -> drivers folder. It should then install the driver. Once it is finished go into properties and notice which COM port it is connected to. You will need this information.

Next, open Visual Studio and find where you downloaded the Serial Gio. You will want to open the SerialGio.sln file. Once that is open, find the file "Configuration.h" (SerialGio -> SerialGio -> Configuriation.h) and change the part in bold ("COM3") #define PLEO_PORT "**COM3**" from "COM3" to whatever to whatever COM port your Pleo is using. Then, go back to the main.cpp and hit the little green triangle in the top menu to compile and run the code.

If everything works you should see a terminal window. Type the command "sound play moo". If Pleo makes a moo sound you are good to go!

Integrating the Pleo into Education

Pleo in the Classrooom

Preschool and K-1st

For the earlier ages, Pleo can be incorporated as a supportive tool and mentor in the classroom. Pleo's durable exterior makes him ideal for interaction with children, but the children should be at an age where they understand that "hurting" the Pleo is a bad thing.

One possible interesting classroom use could be a spelling/learning-to-write activity for the students. Using a Pleo robot modified with a web cam, students could potentially be able to write a word on a piece of paper that the Pleo is able to recognize. If the spelling of the world is correct, Pleo could play a positive noise. If the Pleo is not able to recognize the word that the student has written, Pleo can make a sad or negative noise indicating to the student that what they've written is incorrect. This kind of classroom activity engages the students in a interactive way, and gives them a more personalized experience than what they might normally receive from a classroom teacher.

Pleo can also be used in children at this young age to teach emotional perception. Pleo's ranges of behavior- from happy, to playful, to sick- represent a wide spectrum of emotion that kids do cover in their introductory curriculum in preschool. Students could be given a Pleo to interact with, and be asked to identify what behavior Pleo is exhibiting, and what could've caused the Pleo to go start exhibiting said behavior. Such interaction with Pleo could also be used in interactions with autistic children at a later age- autistic children are shown to be comfortable around robots, and the easy-to-understand emotions of Pleo could provide them with a useful learning tool to see how one's interaction can lead to certain behaviors.

Elementary School

In the middle ages of childhood, incorporating the Pleo into education because more tricky because the older that the children get, the less likely they will be to see Pleo as an authoritative figure. Students will be more likely to be interested in tinkering with Pleo and seeing how he "works" than necessarily playing games or learning emotions from him as might the younger children. Therefore, some more hands-on interaction could be necessary.

Pleo can be used as a valuable tool in learning multiplication and division in the classroom. If Pleo were to be incorporated into mixed-realities game, Pleo could respond to the student's performance as they would play a mathematics game with Pleo as the character within the game as well. Pleo could react to Pleo's performance within the game. This could be accomplished by modifying Pleo with a Bluetooth receptor, and utilizing the method mentioned earlier in the resources of connecting Pleo to a game that can be played via Android. The method can be modified to work with other devices, but the concept of the Pleo interacting with a virtual game is definitely something to be used in the classroom.

Understanding the hardware aspects of the Pleo can also be interesting to elementary school-aged children. At this point, the Pleo can be de-skinned and used to demonstrate how the sensors work. The teacher can connect Pleo to the computer and run a program on the Pleo that will make Pleo play a certain sound every time that a different sensor is touched. Such a lesson can be useful in demonstrating certain concepts in science, such as reflexes and the nervous system. Students can also examine the hardware and have a class discussion regarding how Pleo moves and works.

Pleos can also be incorporated in lessons regarding life and what makes a creature alive or not alive. In elementary school-level science classes, students need to be able to learn how to describe what makes something living or not living. The Pleo can be an example of something that exhibits similar behaviors to something that is living- namely, the ability to convey emotions and respond to interaction in an appropriate fashion. This can be used to foster discussion relating to the concept of life, and will be sure to be an engaging class session for all students involved.

Middle School/High School

When a student has reached a certain level where they can understand how technology and computers work, students are finally ready to start working with the Pleo on a more programming-intensive level. These possibilities can range anywhere from plugging Pleo in and using sound commands and movement commands to make Pleo act a certain way, to writing your own commands in Pawn for those students who are more aware of what they are doing in regards to the Pleo.

The Pleo comes with a variety of movements and sounds that it can make out-of-the-box when connected to the Terminal through the USB cable. Students who are beginning with programming can connect their Pleo through the serial port and then type in a designated set of commands to watch Pleo perform the action that they type in. This is a very beginner approach to Pleo programming, however, it is a good first step for students who have never done any programming before. Inserting a blank SD card into Pleo may bring about a less interrupted command stream, however, students might find an entirely-motionless-except-for-when-commanded Pleo to be eerie to work with. It is completely at the instructor's

discretion to decide how to go about the activity, but it is a great activity for introducing students to interacting with robots on a more technical level.

Programming the Pleo in Pawn is a task more suited for high school students. There are a great many resources on the internet regarding projects that instructors may choose to do with their students and Pleo which involve programming. The best available resources are currently in the ICE Distance Education portal (http://ice-web.cc.gatech.edu/dl/?q=node/66), and include a variety of projects ranging from beginner-level to the advanced level. These projects do require a good knowledge of the computer, so teachers who hope to utilize these projects should be ready to help their students with any technical issues or dilemmas that they might come across.

Human/Robot Interaction

A Spectrum of Reactions

Perhaps the most unpredictable aspect of research in robotics is the initial human reaction to the robot. What differentiates a robot from a regular object is its ability to, not only move on its own (something that a regular object can't do), but to interact with a human via reaction to touch or sound. How does one treat an object that possesses lifelike qualities, can emulate intelligence and emotion, but does not breathe or feel? This post attempts to answer this question of Human/Robot Interaction in regards to the Pleo.

Initial Reactions: How intuitive is Human/Robot Interaction?

Imagine that you are walking down the street and you see an obviously non-living material object walking along the sidewalk on its own… how do you react; is human/robot interaction intuitive or do humans approach robots with caution? Most interactions observed between the Pleo and humans on first encounters indicated that most humans will carefully approach the robot with an inquisitive expression, rather than jump right into physical interaction.

College Students

When bringing the Pleo home from research meetings, I like to bring it out to let my all college-aged housemates interact with the robot. I observed two different initial approaches. If I did not start off by introducing the Pleo as my robot, the question everyone asked upon seeing it was, "What is that?" In these cases, the human would maintain a safe distance from the Pleo until given further explanation. But, if I preceded the showing of the Pleo with an encouraging statement – such as, "Want to see my robot?" – the question and physical reaction I would receive in response would be something like, "What does it do?" and a more relaxed approach to the robot.

The female reaction to the robot is rather different from the male reaction. Girls seem to have a more caring connection to the play and show more affection when interacting with the robot. Boys tend to lose interest relatively quickly after seeing the robot for the first time. Perhaps children will spend more time with the Pleo and get bored less slowly, if the Pleo was programmed to do more things.

**The next categories are observations taken from a June 29th, 2012 social experiment. The Pleos were placed outside the University of Southern California bookstore. Anyone was allowed to stop by and interact with the robots.*

Walker By's

Most people that walked by the robot set-up would at least look at the robots with a curious gaze, as if asking, "What is that?" Few people passing by actually came up to inspect and interact with the robot without encouragement. Walker by's were more likely to stop if asked, "Would you like to come play with the robot?" at which point they would cautiously come over and ask what the robot does. This would be followed by a few minutes of tutorial before they physically interacted with the robot.

Parents

Parents appeared to be more eager to understand what the robot did and usually walked over on their own with their children and initiated the conversation with basic questions. Upon hearing the answers they would turn to their kids and summarize what we had just said, as if encouraging them to interact with the robot. Parents who came alone would ask if it was possible to buy the robot for their children. One woman even came back later with her son so that he could interact with the robot.

Children

The first people to stop by our demos were a few groups of children between the ages of five and ten. They ran up to where we were sitting by the bookstore, having seen PLEO, and asked "Is that a real dinosaur?" When we told them that it was a robotic dinosaur, some asked, "What does it do?" Others lost interest once it was confirmed that it was fake. This indicated to me that without an interest in programming, these children were more interested in the robot as a toy. If they couldn't perceive it as real anymore, they usually didn't care much for it.

Tourists from Kazakhstan

It was particularly interesting to watch the interaction between the Pleos and a group of tourists from Kazakhstan. The group had a large age range and greeted the robot with many emotions – from fear, to curiosity, to attachment. The older girls, perhaps about high school age, didn't pay much attention to the robot after initial introductions, interactions and explanations. The younger boys, perhaps middle school aged, were interested in the robots for quite some time. They asked questions about how to control the robot and the limitations of the interactions. The youngsters of the group were mainly concerned with playing with the robot, one girl even asked for the website where you could buy the Pleo.